マルチメディア通信と分散処理 76－30
（１９９６．５．１７）

三層クライアント／サーバシステムにおける トランザクション処理ミドルウェアの開発

関根徹† 成田秀明† 松本諭† 鈴木勇二††
E-Mail: {toru, narita, matumoto, yuji}@trl.mei.co.jp
†マルチメディアシステム研所
†† AV&CC システム事業センター
松下電器産業株式会社
〒140 東京都品川区東品川 4-5-15 マルチメディアセンター

我々は、三層クライアント / サーバモデルに基づくトランザクション処理ミドルウェアを開発した。ミドルウェアは機能層である中間ノードで稼働し、異種メディア端末の制御を行うエージェント、データベース操作を制御するトレーダ、両者間のメッセージ交換を行うキュー、トランザクションを管理するモニタで構成される。

ミドルウェアは、データベース操作と端末通信の非同期制御、共有メモリを利用したプロセス間高速通信、データベース操作やネットワーク通信を意識させない抽象インタフェースの提供、などを特徴としてライトウエイトに実装されているため、システムの構築が容易で、開発コストを低く抑えられる利点がある。

Development of a Transaction Processing Middleware in the 3-Tiered Client/Server System

Toru SEKINE † Hideaki NARITA † Satoru MATSUMOTO † Yuji SUZUKI ††
† Multimedia Systems Research Laboratory
† AV&CC Systems Integration Business Center
Matsushita Electric Industrial Co., Ltd.
4-5-15 Higashi-Shinagawa, Shinagawa-ku, Tokyo, 140, Japan

We have developed a transaction processing middleware based on the 3-tiered client/server model. This middleware, works on the intermediate node as the functional layer, is constructed with AGENT - controls multimedia terminals, TRADER - controls database accesses, QUEUE - switches messages AGENTs and TRADERs, and MONITOR - manages transactions.

The middleware has typical features, the asynchronous control between database accesses and terminal communications, the high speed inter-process communications with the shared memory, providing abstract interfaces encapsulated database manipulations and network communications, and so on. This light-weight implementation cases system construction easily and could make the cost of development lower.

1 はじめに

WS および PC によるダウンサイジングの広まりは、ハードウェアのコンパクト化のみならず、クライアント / サーバモデルというソフトウェアアーキテクチャの採用が大きな要因となっている。本稿では、次世代アーキテクチャと目される三層クライアント/サーバモデル[1](以降、三層 C/S モデル)を採用したミドルウェアの開発[2]について報告する。

三層 C/S モデルは以下に述べるように、システムを 3 つの層に分け、それぞれが独立した機能ブロックとして協調して動作するシステムアーキテクチャである。その中間層に位置付けられる機能層は、上流のデータ層と下流のプレゼンテーション層の仲介をして重要かつ複雑な処理を行う層であり、システムの設計および実装においては最も検討を要する部分である。本稿で報告するミドルウェアは、この中間層で動作するものであり、多数のプレゼンテーション層ユーザからの要求を制御しながら、データ層の情報を加工し提供するトランザクション処理を主要機能として持つ。

なお、本ミドルウェアは、平成 8 年 4 月にサービスを開始したある自治体の行政情報システムの基盤として稼働している。

2 システムモデル

2.1 三層 C/S モデル

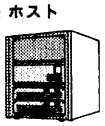
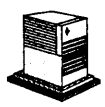
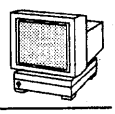
モデル	データ層	機能層	プレゼンテーション層
	・データベース ・既存システム	・アプリケーションロジック ・業務プロセス	・ユーザインタフェース
実体	・メインフレーム ・ホスト	・サーバ WS ・サーバ PC	・WS ・PC
			
計算機世界	リソースマネージャ	サーバ	クライアント

図 1　三層クライアント / サーバモデル

三層 C/S モデルは図 1 に示すように、システムをその目的・機能に着目して、3 つの層に分けるものである。あるまとまったデータを統合管理するデータ層、サービス (機能) の集合体でありプレゼンテーション層からの要求をデータ層の情報を使い処理する機能層、主にユーザインタフェースをつかさどるプレゼンテーション層、で構成される。

これを一般的な計算機世界に投影すると、任意の論理的サービスが情報処理を行なう際の元となる情報 (データ) を管理するリソースマネージャ (通常データベースが適用される)、必要なデータをリソースマネージャから取得し任意の処理を施しまたは付加価値を付与した上でクライアントに提供するサーバ、そしてエンドユーザに直接サービスを提供するクライアントの 3 つのコンポーネントで構成される。

この三層 C/S モデルを採用するシステムでは、一般に以下のような特徴および効果が得られる。

- **柔軟性**
  プロセスから独立したデータ、業務に密接に対応したプロセス、プロセスフローを自由にカスタマイズ可能なユーザインタフェース、など各層で機能が独立しているため、それぞれを他の層に影響を与えず実装したり、任意の組み合わせによりアプリケーションを自由に構築できる
- **拡張性**
  柔軟性での特徴により、各層の範ちゅうで拡張が自由にできる
- **開発効率**
  層ごとに独立しているため、データの共有化、プロセスの再利用が可能である

しかしながら、上記のような確固としたモデルとして実装するためには、データ層におけるデータ構造の設計、機能層で実装する機能の切り出し、プレゼンテーション層でのプロセスフロー定義、など各層での機能分担の明確化が必須となり、それがシステム構築時の一番の課題でもある。さらに、この課題の解には普遍性はなく、適用システム、つまり業務アプリケーションの用途に応じて様々な方法論が存在する。

2.2 機能モデル

我々は、三層C/Sモデルを実システムに適用するべく、具体的ターゲットを定め、そのターゲットのシステム要件を満足するモデルの検討を行なった。ターゲットシステムとして、施設・イベント案内予約系のシステムを取り上げた。ユーザからの要求は、さまざまな端末から大量かつ並列的に発生し、またシステム管理者によるサービス業務の要求も多く発生する。

このようなシステムでの技術課題としては、以下のような点が上げられる。

- **要求の集中および並列化**
  ユーザからの要求トラフィックは大きい。この要求がデータを管理する装置に集中した場合、レスポンスの低下やシステム資源の涸渇につながる。限られた資源の中、パフォーマンスをどう維持するか
- **異機種端末の収容**
  システムが収容する端末種別は多様であり、それぞれプロトコルやユーザインタフェースが異なる。このような異機種端末の差異をどのように吸収し、統一的な設計を行うか
- **アプリケーション開発コスト**
  一般的に大規模システムの開発では、業務アプリのみならずネットワークやデータベースなど広範囲な技術が要求される。これは開発者のスキルに依存することが大きく、いかにこの要因を少なく抑え開発コストを下げるか
- **システムの拡張性**
  システムは、ユーザ数、端末数/種別、およびサービス種別の増加に伴い拡張される。この場合、システム全体に影響を及ぼすことなく、いかに容易に拡張を行うか

我々は、上記の課題を解決する方法として、トランザクション処理ミドルウェアを開発した。図2は、本トランザクション処理ミドルウェアの機能モデルを表したものである。

本モデルの特徴としては、リソースマネージャに対応するコンポーネントとクライアントに対応するコンポーネントを明示的に分離し、それら m 対 n 間でメッセージ交換を行うことである。この端末制御を行うメディア依存部とデータベース操作を行うメディア非依存部が分離されることにより、

- 会話制御に多くの時間を要するクライアントに、データベースコネクションを占有させることがないため、多くのデータベース要求を同時に処理できる
- 異機種クライアントに対し、メディア非依存部で共通した簡易インタフェースを提供することで、その差異を吸収すると共に、開発を容易にする
- クライアントの修正 / 拡張はメディア依存部で、リソースマネージャの修正 / 拡張はメディア非依存部で吸収できるため、システムの保守 / 拡張が容易となる

ことが可能となる。

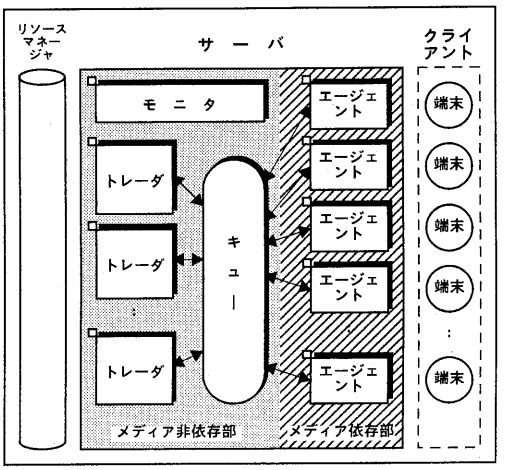


図2　トランザクション処理ミドルウェア

以下に各コンポーネントの概要を説明する。

(1) エージェント

エージェントは、クライアントに一意に対応し、クライアントとの通信、データ変換、フロー制御を行う。端末メディアに依存した処理を行うアプリケーション部分と、トレーダを介したデータベース操作を行う部分の 2 つで構成される。その間に共通インタフェースを設け、アプリケーション開発者にデータベース操作の複雑な作法を隠蔽する。

(2) **トレーダ**

トレーダは、任意のエージェントからの要求を受け取り、リソースマネージャにアクセスを行なう。トレーダは、業務プロセス(アプリケーションロジック)の集合として実装され、エージェントの要求に応じて、適切なプロセスが選択され実行される。

(3) **キュー**

キューは、エージェントとトレーダ間でメッセージ交換を行なう。m対n間で高速にメッセージ交換を行うための排他制御機構および宛先管理機構を持つ。

(4) **モニタ**

モニタはミドルウェア全体の管理に加えて、トランザクション管理機構として、ネームサービス、ロードバランシング、障害検知、そして運用管理機能を持つ。

3 システム構成と機能

我々は機能モデルに基づき、プラットホームをUNIXマシンとし、C言語を用いてミドルウェアの実装を行った。なお、リソースマネージャとしては汎用的なRDBMSを採用した。

以下に、それぞれのコンポーネントごとに構成と機能を説明する。

3.1 エージェントとトレーダ

エージェントおよびトレーダは、アプリケーションロジックを含むUNIXプロセスとして実装される。

(1) **エージェントおよびトレーダの特徴**

- 異機種端末の差異をエージェントで吸収し、端末種別に依存しない共通的なデータベースアクセスをトレーダにより実行できる
- アプリケーション開発者にデータベース操作を隠蔽したインタフェース(業務API)を提供することで、開発効率の向上が期待できる

(2) **業務API**

トレーダは業務プロセスの集合として実装される。エージェントからそのトレーダに対するインタフェースを業務APIと呼び、アプリケーション用途に応じたインタフェースとなっている。このインタフェースは、通常ミドルウェア管理者がアプリケーション用途に応じて、業務プロセスとの対応を取りながら、以降に説明するキューAPIを用いて実装する。

次に、施設案内予約システムで想定される業務APIの例を示す。

auth_user	ユーザ認証
query_fclty	空き施設照会
make_rsrv	施設予約
free_rsrv	予約キャンセル
temp_rsrv	仮予約

(3) **エージェントとトレーダの動作**

図3は、エージェントとトレーダ間でのデータ交換の流れを示したものである。

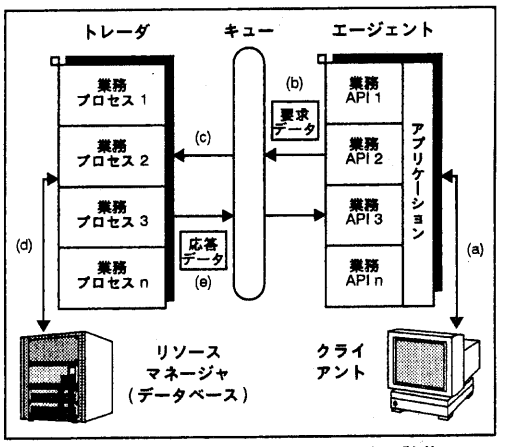


図3　エージェントとトレーダの動作

エージェント内のアプリケーションは、クライアントからの要求を受け取り、端末種別に対応したフロー制御を行いながら、要求内のパラメータと共に関連の業務APIをコールする(a)。業務APIでは、そのパラメータから要求データを作成する。要求データはトレーダ内部でも処理されるため、その構造はエージェントおよびトレーダ間で合意の取れた形となっている。また、業務APIごとに異なった構造となるため、受取側であるトレーダが識別できるよう、内部に要求の種類を示すタグが付加されている。これらは

応答データについても同様である。モニタのネームサービス機能を用いて、要求先のトレーダの宛先を得て、要求データをエンキューする (b)。トレーダは自身への要求データの有無を定期的にポーリングしていて、要求データが存在した場合はそれをデキューする (c)。要求データ内部のタグを解析し、それに対応したデータベース操作を行なう業務プロセスを実行する (d)。データベース操作の結果は応答データとして、反対の経路でクライアントに返される (e)。

3.2 キュー

キューは、異なるプロセス間でデータを蓄積交換する機能をC言語の関数ライブラリ(キューAPI)により提供している。

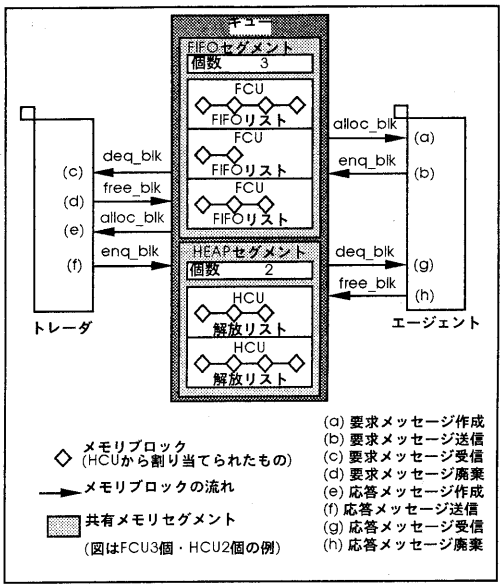


図4　キューの構成

(1) キューの特徴

- キュー API では、プロセス間での排他制御を行なっており、キューに対する複数プロセスの同時アクセスが起こっても矛盾なく処理できる
- 取り扱う要求メッセージや応答メッセージが可変長でも扱えるように、メッセージの長さはキューの持つリソースの制限の範囲内で任意の長さを許している
- キュー API は、SystemV 系 UNIX オペレーティングシステムの提供するプロセス間通信 (IPC) ファシリティを用いて実装しているため、多くのプラットフォーム上で動作可能である

(2) キュー API

キューAPIによるプロセス間のメッセージのやり取りは、基本的に、任意のHEAP制御ユニット(HCU)から割り当てられたメモリブロックにメッセージを入れ、指定した FIFO 制御ユニット(FCU)を介してやり取りする形で行なわれる。HCU はヒープの役割を果たす単位であり、FCU は一般に言う「キュー」という単位である。

次に、キューAPIの主な関数を示す。

alloc_fcu	FCU の割り当て
free_fcu	FCU の解放
alloc_blk	HCU からのメモリブロックの割当
free_blk	HCU へのメモリブロックの解放
enq_blk	FCU へのメッセージの追加
deq_blk	FCU からのメッセージの取得

(3) キューの構成

キューは、SystemV 系 UNIX オペレーティングシステムの提供するIPCファシリティである共有メモリとセマフォを利用して構成される。IPCファシリティとしては、前述の2つの他にメッセージキューがあるが、今回は利用していない。それは、SystemVメッセージキューは、基本的に固定長メッセージの交換において効果的なものであり、可変長メッセージを交換する場合には、データの取り出し操作の繰り返しが発生する可能性があるという点で不利と判断したからである。

キューは、図4に示すように2つの共有メモリセグメント、HEAP セグメントと FIFO セグメントから構成される。キューでは、共有メモリセグメントに保持する情報は、仮想メモリアドレスに依存していないため、これら両セグメントは任意の仮想メモリアドレスにアタッチ可能となっている。これにより、プロセスのメモリ管理上の制約を少なくしている。

3.3 モニタ

モニタは、ミドルウェアの構成要素の管理を行うと共に、トランザクション管理を行う。

(1) **モニタの特徴**

・基本的な TP モニタ機能を有することで、トランザクショナルなアプリケーション用途に対応できる
・ミドルウェア全体の管理を行い、各コンポーネント間の協調動作をスムーズにする

(2) **モニタの主要機能**

次に、モニタの主要機能を示す。

・**システム管理機能**
コンフィギュレーションに応じたシステムを起動時に構成し、ユーザからの要求により稼働時に再構成する
・**ネームサービス機能**
コンポーネント間の通信、データベース操作に必要な情報を通知する
・**状態通知機能**
各コンポーネントの状態およびトランザクションの状態を通知する
・**統計記録機能**
各コンポーネントの統計情報およびトランザクションの統計情報を記録する
・**障害検知機能**
各コンポーネント間でのメッセージ交換時に、相手の正常性を確認する

4 応用システム

図5は、今回開発したトランザクション処理ミドルウェアを採用した、ある自治体のスポーツ施設案内予約システムの構成概要である。本システムでは、従来手作業で行っていたスポーツ施設の案内予約業務が、自宅や勤務先でオンラインで行うことができる。システムは、リソースマネージャとして Oracle7 をデータベースとする業務処理ホスト、サーバとして端末種別ごとに存在する制御装置、クライアントとしてユーザ端末である電話、FAX、パソコン通信端末に加え、駅構内など公共エリアに設置される街頭端末、自治体職員が管理業務に使用する Windows ベースの業務端末、で構成される。

今回、本ミドルウェアを採用することで、レスポンスを低下させることなく多数ユーザからの同時並行要求の処理、異機種端末の収容を、開発コストを最低限に抑え、将来の拡張性を維持しながら実現できた。

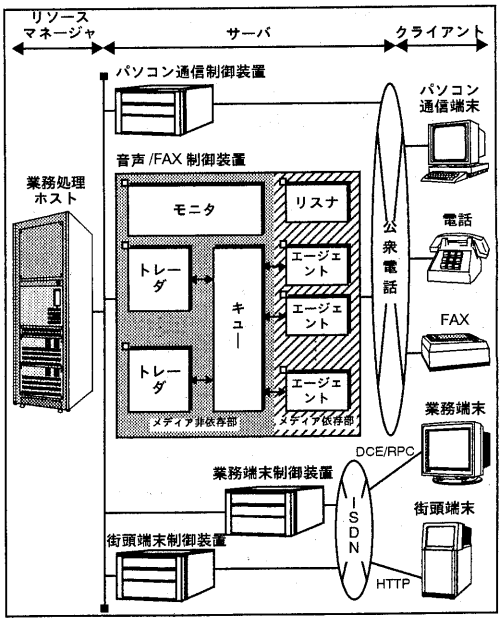


図5　応用システム

5 まとめ

三層 C/S モデルを用いたトランザクション処理ミドルウェアを開発し、その有効性を応用システムに適用することで確認した。今後は、「ライトウェイト」というキーワードを保持しながら、障害対応機能の実装、分散リソースマネージャ対応などを行っていく予定である。

参考文献

[1] John J. Donovan, "Business Re-engineering with Information Technology", Prentice Hall, ISBN 0-13-125907-5

[2] 鈴木 他，「三層クライアント / サーバシステムにおけるミドルウェアの開発 (1)～(5)」，情報処理学会第 51 回全国大会講演論文集 (1), pp.185-194, 1995